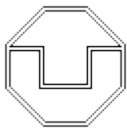


Description Logic

Franz Baader

Literature:

- Franz Baader, Carsten Lutz, Ian Horrocks, and Uli Sattler: **An Introduction to Description Logic**. Cambridge University Press, 2017.
- F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (ed.): **The Description Logic Handbook**. Cambridge University Press, 2003.
- F. Baader, C. Lutz: **Description Logics**. In The Handbook of Modal Logic, Elsevier, 2006.
- F. Baader: **Description Logics**. In Reasoning Web: Semantic Technologies for Information Systems, 5th International Summer School 2009, Springer LNCS 5689, 2009.



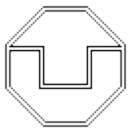
Chapter 1: Introduction

Description Logic

subfield of knowledge representation,
which is a subfield of Artificial Intelligence

Description comes from **concept description**, i.e., a formal expression that determines a set of individuals with common properties

Logics comes from the fact that the **semantics** of concept descriptions can be defined using **logic**;
in particular, most Description Logics can be seen as **fragments of first-order logic**.



Description Logic

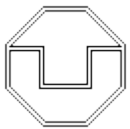
subfield of knowledge representation,
which is a subfield of Artificial Intelligence

Description Logic: name of a research field

Description Logics: a family of knowledge representation languages

Description Logic: a member of this family

DL(s)



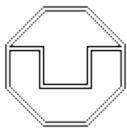
Knowledge Representation

general goal

“develop formalisms for providing high-level descriptions of the world that can be effectively used to build intelligent applications”

[Brachman & Nardi, 2003]

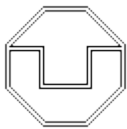
- **formalism:** well-defined **syntax** and formal, unambiguous **semantics**
- **high-level description:** only relevant aspects represented, others left out
- **intelligent applications:** must be able to reason about the knowledge, and infer implicit knowledge from the explicitly represented knowledge
- **effectively used:** need for practical reasoning tools and efficient implementations



Syntax

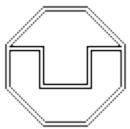
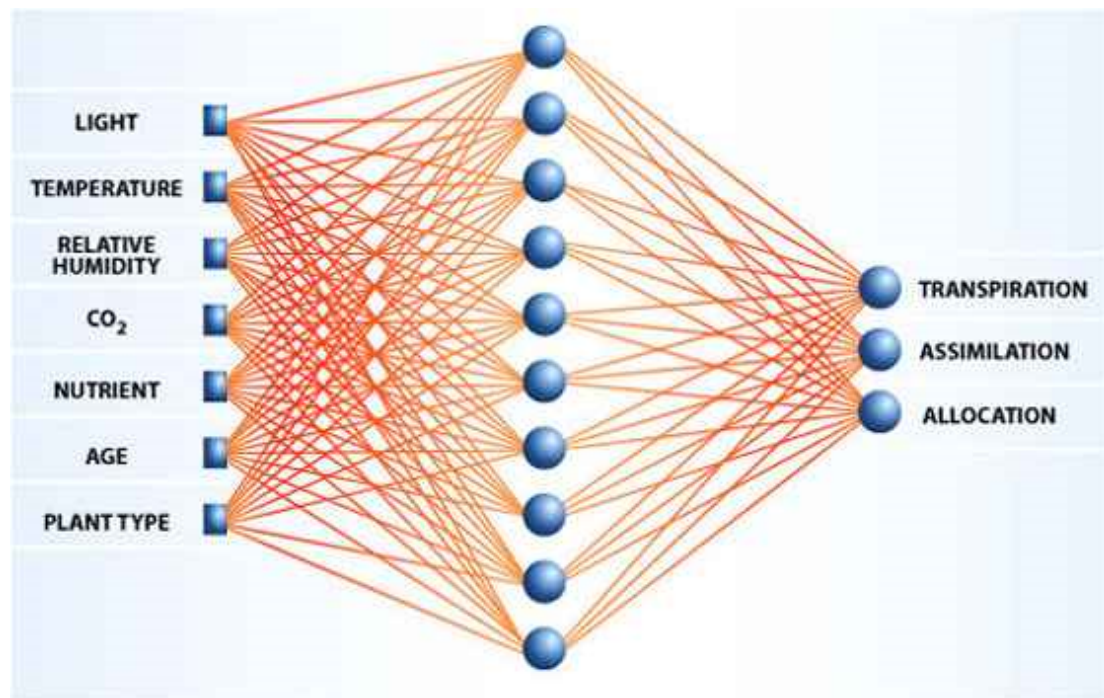
- provides an explicit symbolic representation of the knowledge
- not just implicit, as e.g. in neural networks

Woman	$\equiv$ Person $\sqcap$ Female	
Man	$\equiv$ Person $\sqcap$ $\neg$ Female	
Mother	$\equiv$ Woman $\sqcap$ $\exists$ hasChild. $\top$	
Person	$\equiv$ Man $\sqcup$ Woman	Male(JOHN)
$\perp$	$\equiv$ Male $\sqcap$ Female	Male(MARC)
		Male(STEPHEN)
hasChild(STEPHEN , MARC)		Male(JASON)
hasChild(MARC , ANNA)		Female(MICHELLE)
hasChild(JOHN , MARIA)		Female(ANNA)
hasChild(ANNA , JASON)		Female(MARIA)



Syntax

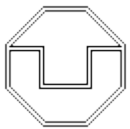
- provides an **explicit symbolic representation** of the knowledge
- **not** just implicit, as e.g. in neural networks



Semantics

describes the connection between the **symbolic representation** and the **real-world entities** it is supposed to represent

- **no procedural semantics**, i.e., should **not** just be defined by how certain programs using the symbolic representation behave
- instead **declarative semantics**:
 - mapping of the symbolic expressions to an abstraction of the “world” (**interpretation**)
 - notion of “**truth**” that allows us to determine whether a symbolic expression is true in the world under consideration (**model**)

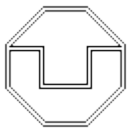


Syntax & Semantics

determine the **expressive power** of a formalism

Adequate expressive power:

- **not too low:** can all the knowledge relevant for solving the problem at hand be represented?
- **not too high:** are the available representational means really necessary in this application?



Reasoning

deduce implicit knowledge from the explicitly represented knowledge

$$\forall x.\forall y. (male(y) \wedge \exists z.(has_child(x, z) \wedge has_child(z, y)) \rightarrow has_grandson(x, y))$$

has_child(John, Mary)

has_child(Mary, Paul)

male(Paul)

grandson_of(John, Paul)

implicit knowledge

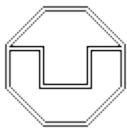
Knowledge representation systems

should provide their users with inference tools

that can deduce (certain) implicit consequences

results should **depend only on the semantics** of the representation language,
and **not on the syntactic representation**:

semantically equivalent knowledge should lead to the same result



Reasoning

deduce implicit knowledge from the explicitly represented knowledge

$\forall x.\forall y.\forall z. (has_child(x, z) \wedge has_child(z, y) \wedge male(y)) \rightarrow has_grandson_of(x, y)$

has_child(John, Mary)

has_child(Mary, Paul)

male(Paul)

grandson_of(John, Paul)

implicit knowledge

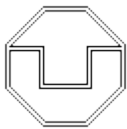
Knowledge representation systems

should provide their users with inference tools

that can deduce (certain) implicit consequences

results should **depend only on the semantics** of the representation language,
and **not on the syntactic representation**:

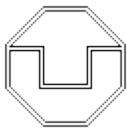
semantically equivalent knowledge should lead to the same result



Reasoning procedures

requirements in knowledge representation

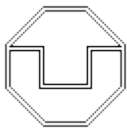
- The procedure should be a **decision procedure** for the problem:
 - **soundness**: positive answers are correct
 - **completeness**: negative answers are correct
 - **termination**: always gives an answer in finite time
- The procedure should be as **efficient** as possible:
preferably **optimal** w.r.t. the (worst-case) complexity of the problem
- The procedure should be **practical**:
easy to implement and optimize, and behave well in applications



Reasoning procedures

example

- Satisfiability in **first-order logic** does not have a decision procedure.
 - full first-order logic is thus not an appropriate knowledge representation formalism
- Satisfiability in **propositional logic** has a decision procedure, but the problem is NP-complete.
 - there are, however, highly optimized **SAT solvers** that behave well in practice
 - **expressive power** is, however, often **not sufficient** to express the relevant knowledge



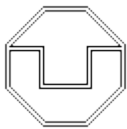
Terminological knowledge

formalize the terminology of the application domain:

- define important notions (classes, relations, objects) of the domain
- state constraints on the way these notions can be interpreted
- deduce consequences of definitions and constraints:
subclass relationships, instance relationships

Example: domain conference

- **classes** (concepts) like **Person, Speaker, Author, Talk, Participant, Workshop, ...**
- **relations** (roles) like **gives, attends, likes, ...**
- **objects** (individuals) like **Richard, Frank, Paper_176, ...**
- **constraints** like: every talk is given by a speaker, every speaker is an author, every workshop must have at least 10 participants, ...

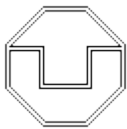


Ontologies

terminological knowledge bases are nowadays often called ontologies

Ontologies are, for example, used in:

- the **Semantic Web** to enable a common understanding of important notions, which can be used in the semantic labeling of Web pages
- **Information Retrieval** to support the automatic extraction of information from text documents
- **Medicine** to provide formal definitions for important notions that can be used by medical doctors to describe findings and procedures, insurance companies to determine payment, ... (SNOMED CT, GALEN, ...)
- **Biology** to enable semantic access to gene databases (Gene Ontology)
- ...

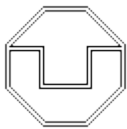


Description Logics

class of logic-based **knowledge representation** formalisms
tailored towards representing **terminological knowledge**

Prehistory:

- Descended from **early approaches** for representing terminological knowledge
 - **semantic networks** (Quillian, 1968)
 - **frames** (Minsky, 1975)
- problems with **missing semantics** lead to
 - **structured inheritance networks** (Brachman, 1978)
 - the first DL system **KL-ONE** (Brachman&Schmolze, 1985)



Description Logic

history

Phase 1:

- implementation of systems (Back, K-Rep, Loom, Meson, ...)
- based on incomplete structural subsumption algorithms

Phase 2:

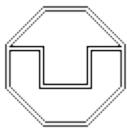
- development of tableau-based algorithms and complexity results
- first implementation of tableau-based systems (Kris, Crack)
- first formal investigation of optimization methods

Phase 3:

- tableau-based algorithms for very expressive DLs
- highly optimized tableau-based systems (FaCT, Racer, HermiT, Konclude, ...)
- relationship to modal logic and decidable fragments of FOL

Phase 4:

- Web Ontology Language (OWL-DL) based on very expressive DL
- industrial-strength reasoners and ontology editors for OWL-DL
- investigation of light-weight DLs with tractable reasoning problems
- query answering w.r.t. ontologies for large data sets



Chapter 2

A Basic Description Logic

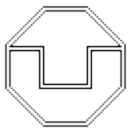
$\mathcal{ALC}$

attributive language with complement

[Schmidt-Schauß&Smolka, 1991]

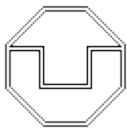
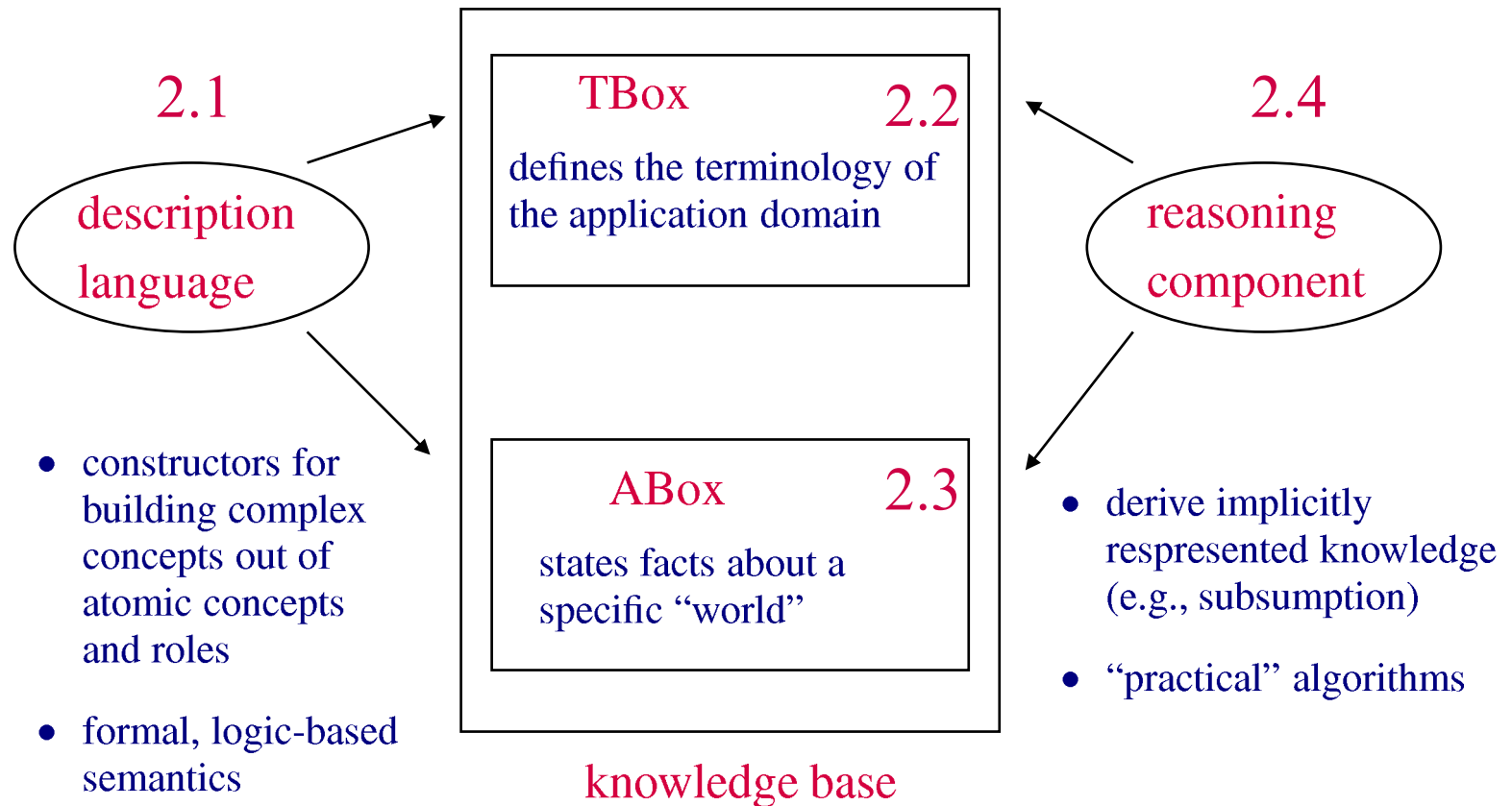
Naming scheme:

- basic language $\mathcal{AL}$
- extended with **constructors** whose “letter” is added after the $\mathcal{AL}$
- $\mathcal{C}$ stands for **complement**, i.e., $\mathcal{ALC}$ is obtained from $\mathcal{AL}$ by adding the complement ($\neg$) operator



Description logic system

structure



Dresden

2.1. The description language

syntax and semantics of $\mathcal{ALC}$

Definition 2.1 (Syntax of $\mathcal{ALC}$)

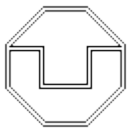
Let $\mathbf{C}$ and $\mathbf{R}$ be disjoint sets of **concept names** and **role names**, respectively.

$\mathcal{ALC}$ -concept descriptions are defined by induction:

- If $A \in \mathbf{C}$, then A is an $\mathcal{ALC}$ -concept description.
- If C, D are $\mathcal{ALC}$ -concept descriptions, and $r \in \mathbf{R}$, then the following are $\mathcal{ALC}$ -concept descriptions:
 - $C \sqcap D$ (conjunction)
 - $C \sqcup D$ (disjunction)
 - $\neg C$ (negation)
 - $\forall r.C$ (value restriction)
 - $\exists r.C$ (existential restriction)

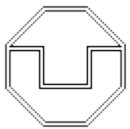
Abbreviations:

- $\top := A \sqcup \neg A$ (top)
- $\perp := A \sqcap \neg A$ (bottom)
- $C \Rightarrow D := \neg C \sqcup D$ (implication)



Notation (use and abuse):

- concept names are called **atomic**
- all other descriptions are called **complex**
- instead of *ALC*-concept description we often say *ALC*-concept or concept description or concept
- *A*, *B* often used for concept names, *C*, *D* for complex concept descriptions, *r*, *s* for role names



The description language

examples of $\mathcal{ALC}$ -concept descriptions

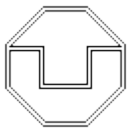
$\text{Person} \sqcap \text{Female}$

$\text{Participant} \sqcap \exists \text{attends.Talk}$

$\text{Participant} \sqcap \forall \text{attends.}(\text{Talk} \sqcap \neg \text{Boring})$

$\text{Speaker} \sqcap \exists \text{gives.}(\text{Talk} \sqcap \forall \text{topic.DL})$

$\text{Speaker} \sqcap \forall \text{gives.}(\text{Talk} \sqcap \exists \text{topic.}(\text{DL} \sqcup \text{FuzzyLogic}))$



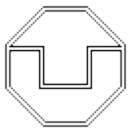
Definition 2.2 (Semantics of $\mathcal{ALC}$)

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an extension mapping $\cdot^{\mathcal{I}}$:

- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in \mathbf{C}$, concepts interpreted as sets
- $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $r \in \mathbf{R}$. roles interpreted as binary relations

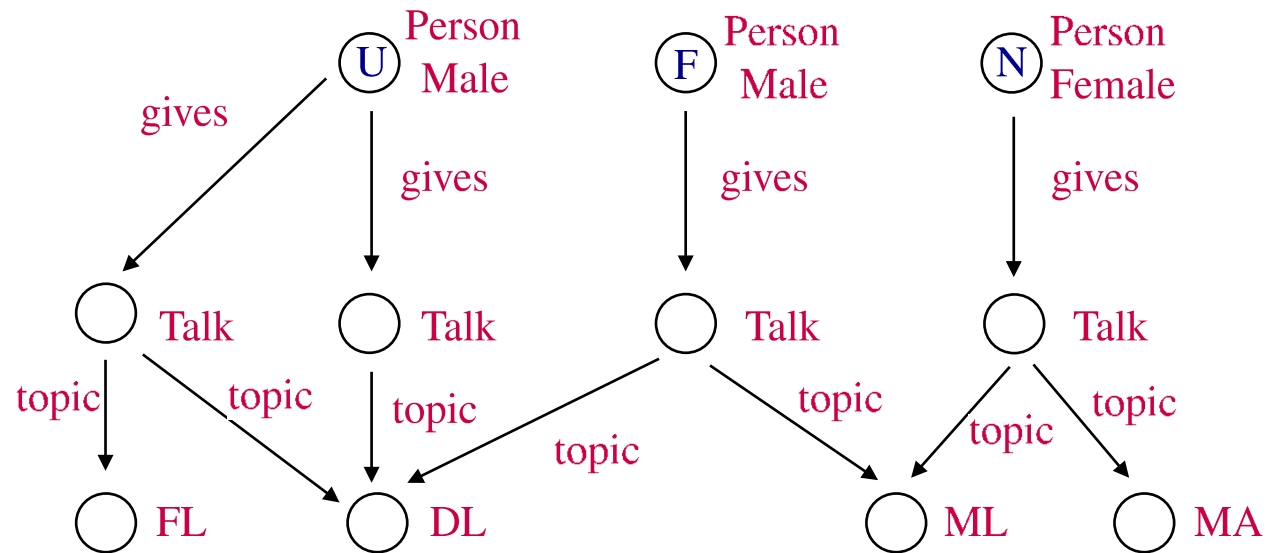
The extension mapping is extended to complex $\mathcal{ALC}$ -concept descriptions as follows:

- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(\forall r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{for all } e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}}\}$
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{there is } e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$



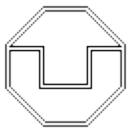
Example

of an interpretation



$\text{Person} \sqcap \exists \text{gives} . (\text{Talk} \sqcap \forall \text{topic} . \text{DL})$

$\text{Person} \sqcap \forall \text{gives} . (\text{Talk} \sqcap \exists \text{topic} . \text{DL})$



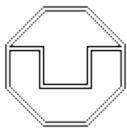
Relationship with First-Order Logic

$\mathcal{ALC}$ can be seen as a fragment of first-order logic:

- Concept names are **unary predicates**, and role names are **binary predicates**.
- **Interpretations** for $\mathcal{ALC}$ can then obviously be viewed as first-order interpretations for this signature.
- Concept descriptions correspond to **first-order formulae with one free variable**.
- Given such a formula $\phi(x)$ with the free variable x and an interpretation $\mathcal{I}$, the **extension** of ϕ w.r.t. $\mathcal{I}$ is given by

$$\phi^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models \phi(d)\}$$

- **Goal:** translate $\mathcal{ALC}$ -concepts C into first-order formulae $\tau_x(C)$ such that their extensions coincide.



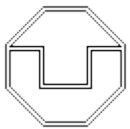
Relationship with First-Order Logic

Concept description C translated into formula with one free variable $\tau_x(C)$:

- $\tau_x(A) := A(x)$ for $A \in \mathbf{C}$
- $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$
- $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$
- $\tau_x(\neg C) := \neg \tau_x(C)$
- $\tau_x(\forall r.C) := \forall y.(r(x, y) \rightarrow \tau_y(C))$
- $\tau_x(\exists r.C) := \exists y.(r(x, y) \wedge \tau_y(C))$

y variable different from x

$$\begin{aligned}\tau_x(\forall r.(A \sqcap \exists r.B)) &= \forall y.(r(x, y) \rightarrow \tau_y(A \sqcap \exists r.B)) \\ &= \forall y.(r(x, y) \rightarrow (A(y) \wedge \exists z.(r(y, z) \wedge B(z))))\end{aligned}$$



Relationship with First-Order Logic

Concept description C translated into formula with one free variable $\tau_x(C)$:

- $\tau_x(A) := A(x)$ for $A \in \mathbf{C}$
- $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$
- $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$
- $\tau_x(\neg C) := \neg \tau_x(C)$
- $\tau_x(\forall r.C) := \forall y.(r(x, y) \rightarrow \tau_y(C))$
- $\tau_x(\exists r.C) := \exists y.(r(x, y) \wedge \tau_y(C))$

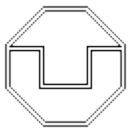
y variable different from x

Lemma 2.3

C and $\tau_x(C)$ have the same extension, i.e.,

$$C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models \tau_x(C)(d)\}$$

Proof: induction on the structure of C



Relationship with First-Order Logic

$\mathcal{ALC}$ can be seen as a fragment of first-order logic:

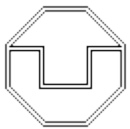
- Concept names are unary predicates, and role names are binary predicates.
- Concept descriptions C yield formulae with one free variable $\tau_x(C)$.

These formulae belong to known decidable subclasses of first-order logic:

- two-variable fragment
- guarded fragment

$$\begin{aligned}\tau_x(\forall r.(A \sqcap \exists r.B)) &= \forall y.(r(x, y) \rightarrow \tau_y(A \sqcap \exists r.B)) \\ &= \forall y.(r(x, y) \rightarrow (A(y) \wedge \exists z.(r(y, z) \wedge B(z))))\end{aligned}$$

$$\begin{aligned}\tau_x(\forall r.(A \sqcap \exists r.B)) &= \forall y.(r(x, y) \rightarrow \tau_y(A \sqcap \exists r.B)) \\ &= \forall y.(r(x, y) \rightarrow (A(y) \wedge \exists x.(r(y, x) \wedge B(x))))\end{aligned}$$

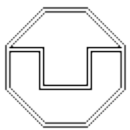


Relationship with Modal Logic

$\mathcal{ALC}$ is a syntactic variant of the basic modal logic K :

- Concept names are **propositional variables**, and role names are names for **transition relations**.
- Concept descriptions C yield **modal formulae** $\pi(C)$:
 - $\pi(A) := a$ for $A \in \mathbf{C}$
 - $\pi(C \sqcap D) := \pi(C) \wedge \pi(D)$
 - $\pi(C \sqcup D) := \pi(C) \vee \pi(D)$
 - $\pi(\neg C) := \neg \pi(C)$
 - $\pi(\forall r.C) := [r]\pi(C)$
 - $\pi(\exists r.C) := \langle r \rangle \pi(C)$

multimodal K :
several pairs of
boxes and diamonds



C and $\pi(C)$ have the **same semantics**: $C^{\mathcal{I}}$ is the set of worlds that make $\pi(C)$ true in the Kripke structure described by $\mathcal{I}$.

Additional constructors

$\mathcal{ALC}$ is only an example of a description logic.

DL researchers have introduced and investigated many additional constructors.

Example

letter Q in the naming scheme

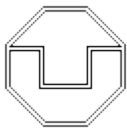
Qualified number restrictions: $(\geq n r.C)$, $(\leq n r.C)$ with semantics

$$(\geq n r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \geq n\}$$

$$(\leq n r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \leq n\}$$

Persons that attend at most 20 talks, of which at least 3 have the topic DL:

$$\text{Person} \sqcap (\leq 20 \text{ attends.Talk}) \sqcap (\geq 3 \text{ attends.}(\text{Talk} \sqcap \exists \text{topic.DL}))$$



Additional constructors

$\mathcal{ALC}$ is only an example of a description logic.

DL researchers have introduced and investigated many additional constructors.

Example

letter $\mathcal{Q}$ in the naming scheme

Qualified number restrictions: $(\geq n r.C)$, $(\leq n r.C)$ with semantics

$$(\geq n r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \geq n\}$$

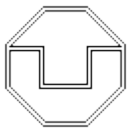
$$(\leq n r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \leq n\}$$

Number restrictions: $(\geq n r)$, $(\leq n r)$ as abbreviation for $(\geq n r.\top)$ and $(\leq n r.\top)$:

$$(\geq n r)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}}\}) \geq n\}$$

$$(\leq n r)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{card}(\{e \mid (d, e) \in r^{\mathcal{I}}\}) \leq n\}$$

letter $\mathcal{N}$ in the naming scheme



Additional constructors

In addition to concept constructors, one can also introduce **role constructors**.

Example

letter $\mathcal{I}$ in the naming scheme

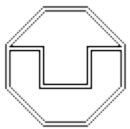
Inverse roles: if r is a role, then r^- denotes its inverse

$$(r^-)^{\mathcal{I}} := \{(e, d) \mid (d, e) \in r^{\mathcal{I}}\}$$

Inverse roles can be used like role names in value and existential restrictions.

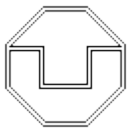
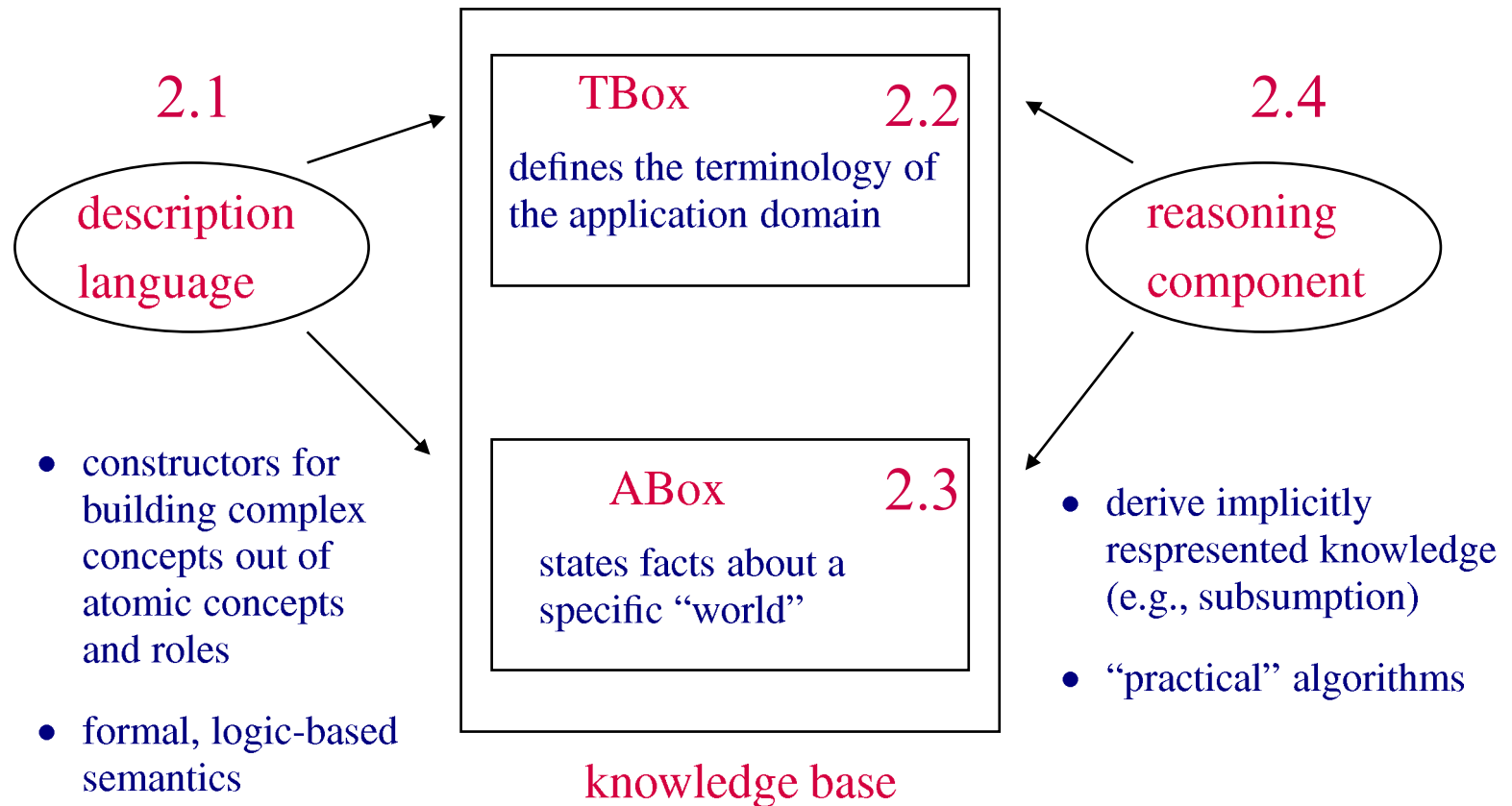
Presenter of a boring talk:

$\text{Speaker} \sqcap \exists \text{gives}. (\text{Talk} \sqcap \forall \text{attends}^-. (\text{Bored} \sqcup \text{Sleeping}))$



Description logic system

structure



Dresden

2.2. Terminological knowledge

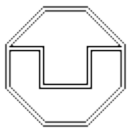
GCI, TBox, and concept definitions

Definition 2.4 (GCI and TBox)

- A **general concept inclusion** is of the form $C \sqsubseteq D$ where C, D are concept descriptions.
- A **TBox** is a finite set of GCIs.
- The interpretation $\mathcal{I}$ **satisfies** the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- The interpretation $\mathcal{I}$ is a **model** of the TBox $\mathcal{T}$ iff it satisfies all the GCIs in $\mathcal{T}$.

Note: this definition is not specific for $\mathcal{ALC}$.

It applies also to other concept description languages.



2.2. Terminological knowledge

GCI, TBoxes, and concept definitions

Definition 2.4 (GCI and TBoxes)

- A **general concept inclusion** is of the form $C \sqsubseteq D$ where C, D are concept descriptions.
- A **TBox** is a finite set of GCIs.
- The interpretation $\mathcal{I}$ **satisfies** the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- The interpretation $\mathcal{I}$ is a **model** of the TBox $\mathcal{T}$ iff it satisfies all the GCIs in $\mathcal{T}$.

$\text{Talk} \sqcap \forall \text{attends}^-. \text{Sleeping} \sqsubseteq \text{Boring}$

$\text{Author} \sqcap \text{PCchair} \sqsubseteq \perp$



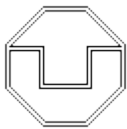
2.2. Terminological knowledge

GCI, TBoxes, and concept definitions

Definition 2.4 (GCI and TBoxes)

- A **general concept inclusion** is of the form $C \sqsubseteq D$ where C, D are concept descriptions.
- A **TBox** is a finite set of GCIs.
- The interpretation $\mathcal{I}$ **satisfies** the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- The interpretation $\mathcal{I}$ is a **model** of the TBox $\mathcal{T}$ iff it satisfies all the GCIs in $\mathcal{T}$.

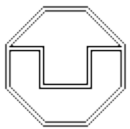
Notation: two TBoxes are called **equivalent** if they have the same models



More GCIs $\Rightarrow$ less models

Lemma 2.5

If $\mathcal{T} \subseteq \mathcal{T}'$ for two TBoxes $\mathcal{T}$, $\mathcal{T}'$, then
each model of $\mathcal{T}'$ is also a model of $\mathcal{T}$.



Restricted TBoxes

concept definitions and acyclic TBoxes

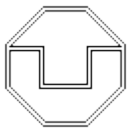
Definition 2.6

A **concept definition** is of the form $A \equiv C$ where

- A is a concept name;
- C is a concept description.

The interpretation $\mathcal{I}$ **satisfies** the concept definition $A \equiv C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$.

↑
abbreviation for the two GCIs
 $A \sqsubseteq C$ and $C \sqsubseteq A$



Restricted TBoxes

concept definitions and acyclic TBoxes

Definition 2.6 (continued)

An **acyclic TBox** is a finite set of concept definitions that

- does **not** contain **multiple definitions**;
- does **not** contain **cyclic definitions**.

multiple definition

$$\begin{array}{l} A \equiv C \\ A \equiv D \end{array} \quad \text{for } C \neq D$$

$$\begin{array}{l} A \equiv B \sqcap \forall r.P \\ B \equiv P \sqcap \forall r.C \\ C \equiv \exists r.A \end{array}$$

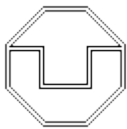
cyclic definition

No cyclic definitions:

there is no sequence $A_1 \equiv C_1, \dots, A_n \equiv C_n \in \mathcal{T}$ ($n \geq 1$) such that

- A_{i+1} occurs in C_i ($1 \leq i < n$)
- A_1 occurs in C_n

Case $n = 1$?



Restricted TBoxes

concept definitions and acyclic TBoxes

Definition 2.6 (continued)

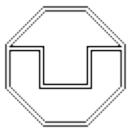
An **acyclic TBox** is a finite set of concept definitions that

- does **not** contain **multiple definitions**;
- does **not** contain **cyclic definitions**.

The interpretation $\mathcal{I}$ is a **model** of the acyclic TBox $\mathcal{T}$ iff it **satisfies all** its **concept definitions**: $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all $A \equiv C \in \mathcal{T}$

Given an acyclic TBox, we call a **concept name** A occurring in $\mathcal{T}$ a

- **defined concept** iff there is C such that $A \equiv C \in \mathcal{T}$;
- **primitive concept** otherwise.



Example

of an acyclic TBox

Woman $\equiv$ Person $\sqcap$ Female

Man $\equiv$ Person $\sqcap$ $\neg$ Female

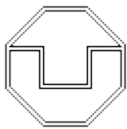
Talk $\equiv$ $\exists$ topic. $\top$

Speaker $\equiv$ Person $\sqcap$ $\exists$ gives.Talk

Participant $\equiv$ Person $\sqcap$ $\exists$ attends.Talk

BusySpeaker $\equiv$ Speaker $\sqcap$ (≥ 3 gives.Talk)

BadSpeaker $\equiv$ Speaker $\sqcap$ $\forall$ gives. ($\forall$ attends $^\neg$. (Bored $\sqcup$ Sleeping))



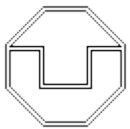
Acyclic TBoxes

an important result

Proposition 2.7

For every acyclic TBox $\mathcal{T}$ we can effectively construct an equivalent acyclic TBox $\hat{\mathcal{T}}$ such that the **right-hand sides** of concept definitions in $\hat{\mathcal{T}}$ contain **only primitive concepts**.

Proof: blackboard



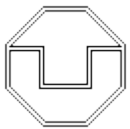
Acyclic TBoxes

an important result

Proposition 2.7

For every acyclic TBox $\mathcal{T}$ we can effectively construct an equivalent acyclic TBox $\hat{\mathcal{T}}$ such that the **right-hand sides** of concept definitions in $\hat{\mathcal{T}}$ contain **only primitive concepts**.

We call $\hat{\mathcal{T}}$ the *expanded version* of $\mathcal{T}$.



Acyclic TBoxes

an important result

Given an acyclic TBox $\mathcal{T}$, a **primitive interpretation** $\mathcal{J}$ for $\mathcal{T}$ consists of a nonempty set $\Delta^{\mathcal{J}}$ together with an extension mapping $\cdot^{\mathcal{J}}$, that maps

- **primitive** concepts P to sets $P^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}$
- role names r to binary relations $r^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$

The interpretation $\mathcal{I}$ is an **extension** of the primitive interpretation $\mathcal{J}$ iff $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$ and

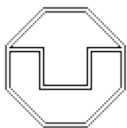
- $P^{\mathcal{J}} = P^{\mathcal{I}}$ for all primitive concepts P
- $r^{\mathcal{J}} = r^{\mathcal{I}}$ for all role names r

Corollary 2.8

Let $\mathcal{T}$ be an acyclic TBox.

Any **primitive interpretation** $\mathcal{J}$ has a **unique extension** to a model of $\mathcal{T}$.

Proof: blackboard



Relationship with First-Order Logic

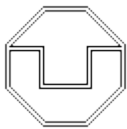
$\mathcal{ALC}$ -TBoxes can be translated into first-order logic:

$$\tau(\mathcal{T}) := \bigwedge_{C \sqsubseteq D \in \mathcal{T}} \forall x. (\tau_x(C) \rightarrow \tau_x(D))$$

Lemma 2.9

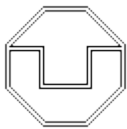
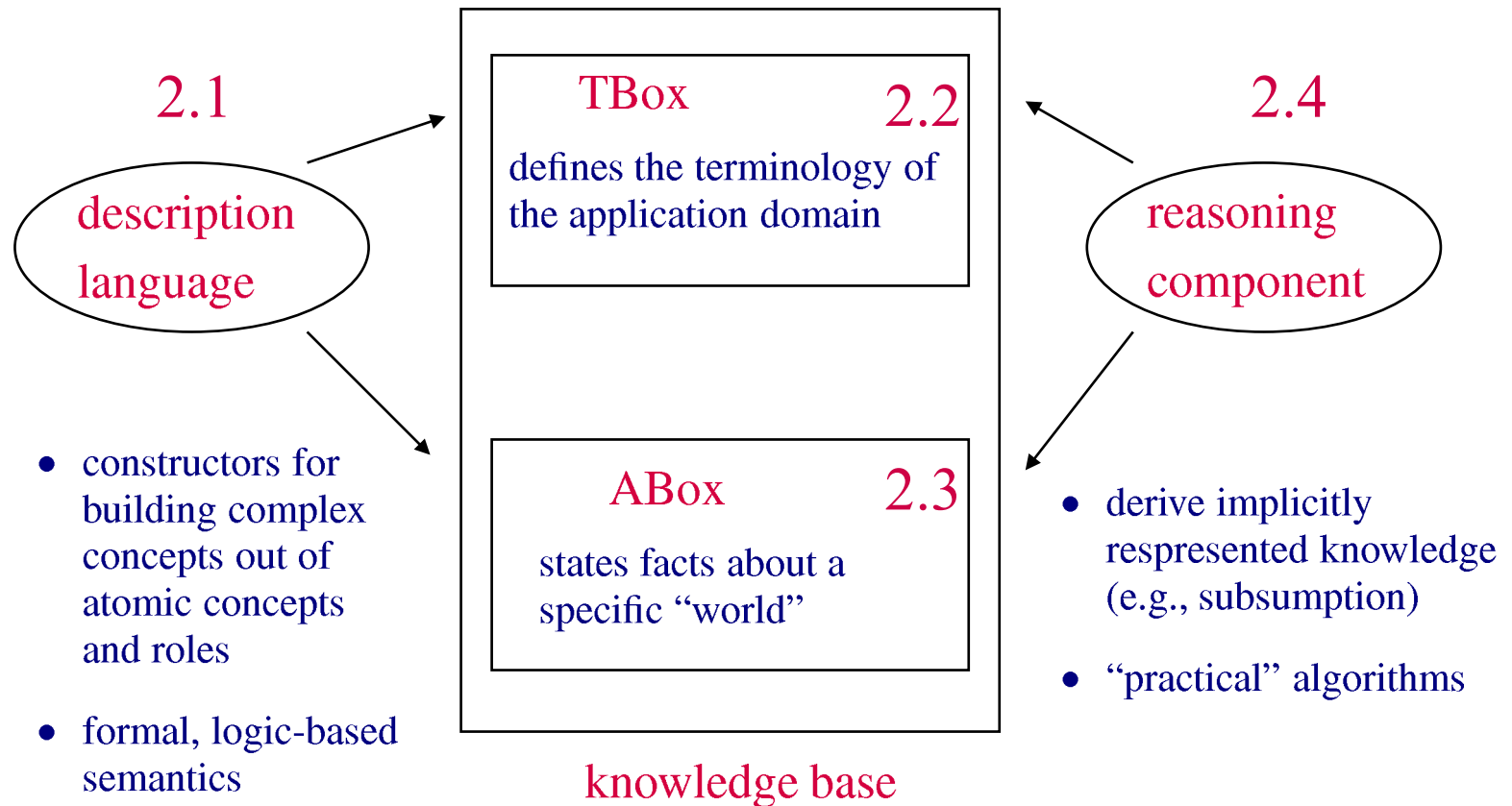
Let $\mathcal{T}$ be a TBox and $\tau(\mathcal{T})$ its translation into first-order logic.
Then $\mathcal{T}$ and $\tau(\mathcal{T})$ have the same models.

Proof: blackboard



Description logic system

structure



Dresden

2.3. Assertional knowledge

Definition 2.10 (Assertions and ABoxes)

An **assertion** is of the form

$$a : C \text{ (concept assertion)} \quad \text{or} \quad (a, b) : r \text{ (role assertion)}$$

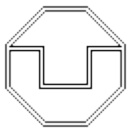
where C is a concept description, r is a role, and a, b are **individual names** from a **set I** of such names (disjoint with **C** and **R**).

An **ABox** is a finite set of assertions.

An interpretation $\mathcal{I}$ is a **model** of an ABox $\mathcal{A}$ if it **satisfies all its assertions**:

$$\begin{array}{ll} a^{\mathcal{I}} \in C^{\mathcal{I}} & \text{for all } a : C \in \mathcal{A} \\ (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} & \text{for all } (a, b) : r \in \mathcal{A} \end{array}$$

$\mathcal{I}$ assigns elements $a^{\mathcal{I}}$
of $\Delta^{\mathcal{I}}$ to individual names
 $a \in \mathbf{I}$



2.3. Assertional knowledge

Definition 2.10 (Assertions and ABoxes)

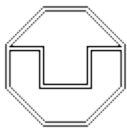
An **assertion** is of the form

$a : C$ (concept assertion) or $(a, b) : r$ (role assertion)

where C is a concept description, r is a role, and a, b are **individual names** from a **set I** of such names (disjoint with **C** and **R**).

An **ABox** is a finite set of assertions.

FRANZ : Lecturer, (FRANZ, TU03) : teaches,
TU03 : Tutorial, (TU03, RinDL) : topic,
RinDL : DL



Relationship with First-Order Logic

$\mathcal{ALC}$ -ABoxes can be translated into first-order logic:

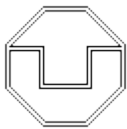
$$\tau(\mathcal{A}) := \bigwedge_{a: C \in \mathcal{T}} \tau_x(C)(a) \wedge \bigwedge_{(a,b): r \in \mathcal{T}} r(a,b)$$

individual names are
viewed as constants

Lemma 2.11

Let $\mathcal{A}$ be an ABox and $\tau(\mathcal{A})$ its translation into first-order logic.
Then $\mathcal{A}$ and $\tau(\mathcal{A})$ have the same models.

Proof: easy



Knowledge Bases

Definition 2.12

A **knowledge base** $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

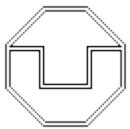
The interpretation $\mathcal{I}$ is a **model** of the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$.

First-order translation: $\tau(\mathcal{K}) := \tau(\mathcal{T}) \wedge \tau(\mathcal{A})$

Lemma 2.13

Let $\mathcal{K}$ be a knowledge base and $\tau(\mathcal{K})$ its translation into first-order logic. Then $\mathcal{K}$ and $\tau(\mathcal{K})$ have the **same models**.

Proof: immediate consequence of Lemma 2.9 and Lemma 2.11



Additional constructors

Individual names can also be used as **concept constructors** to increase the expressive power of the concept description language.

They yield a **singleton set** consisting of the extension of the individual name.

Nominals

letter $\mathcal{O}$ in the naming scheme

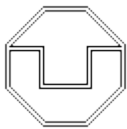
Nominals: $\{a\}$ for $a \in \mathbf{I}$ with semantics

$$\{a\}^{\mathcal{I}} := \{a^{\mathcal{I}}\}$$

Nominals can be used to express ABox assertions using GCIs:

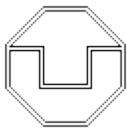
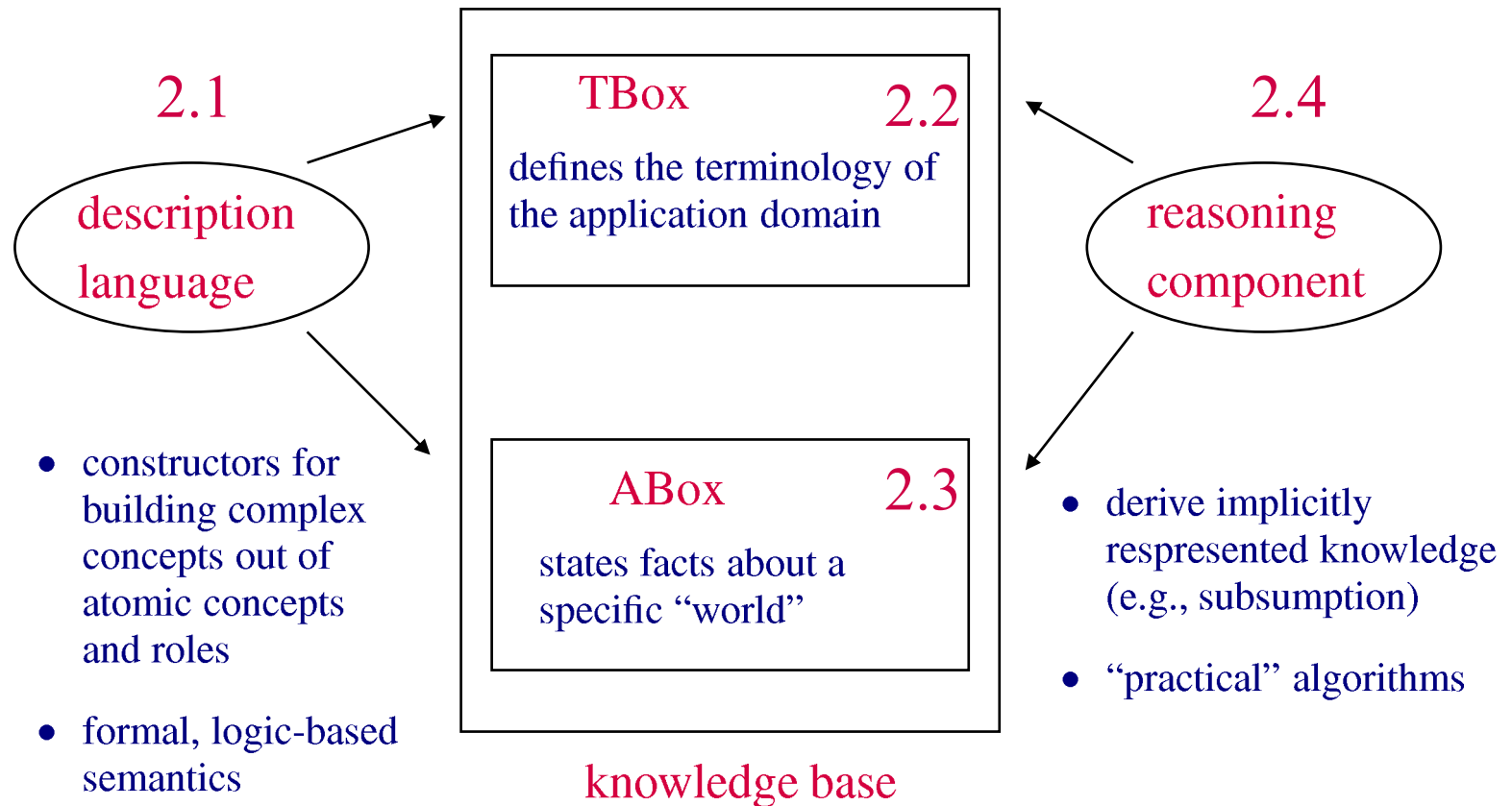
$a : C$ is expressed by $\{a\} \sqsubseteq C$

$(a, b) : r$ is expressed by $\{a\} \sqsubseteq \exists r. \{b\}$



Description logic system

structure



Dresden

2.4. Reasoning Problems and Services

make implicitly
represented
knowledge explicit

Definition 2.14 (terminological reasoning) Let $\mathcal{T}$ be a TBox.

TBox consistency:

$\mathcal{T}$ is **consistent** iff it has a model.

Satisfiability:

C is **satisfiable** w.r.t. $\mathcal{T}$ iff $C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$ of $\mathcal{T}$.

Subsumption:

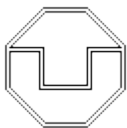
C is **subsumed** by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff

$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of the TBox $\mathcal{T}$.

Equivalence:

C is **equivalent** to D w.r.t. $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) iff

$C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of the TBox $\mathcal{T}$.



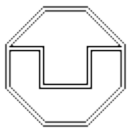
Terminological Reasoning

Note:

If $\mathcal{T} = \emptyset$, then satisfiability/subsumption/equivalence w.r.t. $\mathcal{T}$ is simply called satisfiability/subsumption/equivalence and we write $\sqsubseteq$ and $\equiv$.

Examples:

- $A \sqcap \neg A$ and $\forall r.A \sqcap \exists r.\neg A$ are not satisfiable (unsatisfiable)
- $A \sqcap \neg A$ and $\forall r.A \sqcap \exists r.\neg A$ are equivalent
- $A \sqcap B$ is subsumed by A and by B .
- $\exists r.(A \sqcap B)$ is subsumed by $\exists r.A$ and by $\exists r.B$
- $\forall r.(A \sqcap B)$ is equivalent to $\forall r.A \sqcap \forall r.B$
- $\exists r.A \sqcap \forall r.B$ is subsumed by $\exists r.(A \sqcap B)$



Properties of Subsumption

Lemma 2.15

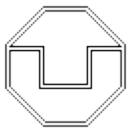
- The subsumption relation $\sqsubseteq_{\mathcal{T}}$ is a **pre-order** on concept descriptions, i.e.,

- $C \sqsubseteq_{\mathcal{T}} C$ (reflexive)
- $C \sqsubseteq_{\mathcal{T}} D \wedge D \sqsubseteq_{\mathcal{T}} E \rightarrow C \sqsubseteq_{\mathcal{T}} E$ (transitive)

It is **not a partial order** since it is not antisymmetric:

- $C \sqsubseteq_{\mathcal{T}} D \wedge D \sqsubseteq_{\mathcal{T}} C \not\rightarrow C = D$
- The constructors **existential restriction** and **value restriction** are **monotonic** w.r.t. subsumption, i.e.,
 - $C \sqsubseteq_{\mathcal{T}} D \rightarrow \exists r.C \sqsubseteq_{\mathcal{T}} \exists r.D \wedge \forall r.C \sqsubseteq_{\mathcal{T}} \forall r.D$
- **Subsumption reasoning** is **monotonic**, i.e., if $\mathcal{T} \subseteq \mathcal{T}'$, then
 - $C \sqsubseteq_{\mathcal{T}} D \rightarrow C \sqsubseteq_{\mathcal{T}'} D$

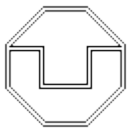
Proof: blackboard



Basic equivalences

Lemma 2.16

- $\neg \top \equiv \perp$
- $C \sqcup D \equiv \neg(\neg C \sqcap D)$
- $\forall r.C \equiv \neg \exists r.\neg C$
- $\exists r.\neg \perp \equiv \perp$
- $\neg(\geq n r.C) \equiv (\leq n - 1 r.C)$ if $n \geq 1$
- $(\geq 0 r.C) \equiv \top$
- $(\leq 0 r.C) \equiv \forall r.\neg C$



Assertional Reasoning

Definition 2.17 (assertional reasoning)

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base.

Consistency:

$\mathcal{K}$ is **consistent** iff there exists a model of $\mathcal{K}$.

Instance:

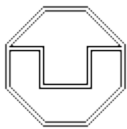
a is an **instance of** C w.r.t. $\mathcal{K}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$.

Lemma 2.18

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base.

If a is an instance of C w.r.t. $\mathcal{K}$ and $C \sqsubseteq_{\mathcal{T}} D$,
then a is an instance of D w.r.t. $\mathcal{K}$.

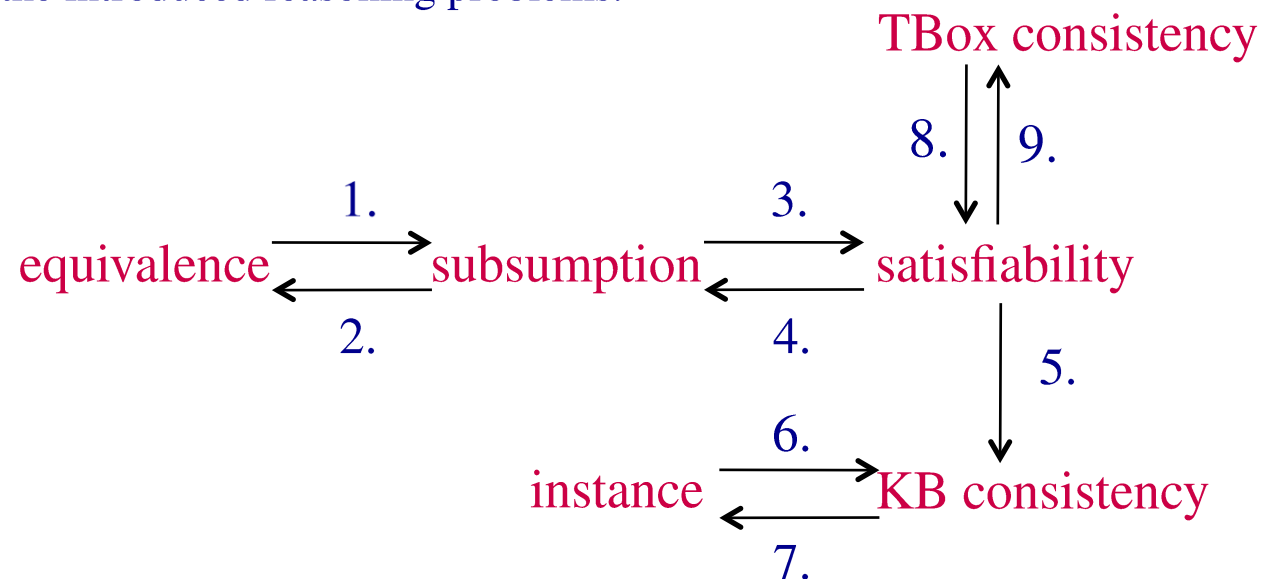
Proof: exercise



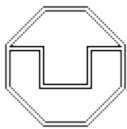
Reductions

between reasoning problems

There are the following **polynomial time** reductions between the introduced reasoning problems:



This holds not only for $\mathcal{ALC}$, but for all DLs that have the constructors **conjunction** and **negation**.



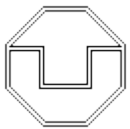
Theorem 2.19

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, C, D concept descriptions, and $a \in \mathbf{I}$.

1. $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$
2. $C \sqsubseteq_{\mathcal{T}} D$ iff $C \equiv_{\mathcal{T}} C \sqcap D$
3. $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$
4. C is satisfiable w.r.t. $\mathcal{T}$ iff $C \not\sqsubseteq_{\mathcal{T}} \perp$
5. C is satisfiable w.r.t. $\mathcal{T}$ iff $(\mathcal{T}, \{a : C\})$ is consistent
6. a is an instance of C w.r.t. $\mathcal{K}$ iff $(\mathcal{T}, \mathcal{A} \cup \{a : \neg C\})$ is inconsistent
7. $\mathcal{K}$ is consistent iff a is not an instance of $\perp$ w.r.t. $\mathcal{K}$
8. $\mathcal{T}$ is consistent iff $\top$ is **satisfiable** w.r.t. $\mathcal{T}$
9. C is satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \cup \{\top \sqsubseteq \exists r.C\}$ is consistent

Proof: blackboard

↑
new role



Reduction

getting rid of acyclic TBoxes

Expansion of concepts and ABoxes:

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, where $\mathcal{T}$ is acyclic, and C a concept description.

The expanded versions $\hat{C}$ and $\hat{\mathcal{A}}$ of C and $\mathcal{A}$ w.r.t. $\mathcal{T}$ are obtained as follows:

- replace all defined concepts occurring in C and $\mathcal{A}$ by their definitions in the expanded version $\hat{\mathcal{T}}$ of $\mathcal{T}$.

$\mathcal{T}$

Woman	$\equiv$	Person $\sqcap$ Female
Talk	$\equiv$	$\exists \text{topic}.\top$
Speaker	$\equiv$	Person $\sqcap \exists \text{gives}.\text{Talk}$

$C = \text{Woman} \sqcap \text{Speaker}$ expands to

$\hat{C} = \text{Person} \sqcap \text{Female} \sqcap \text{Person} \sqcap \exists \text{gives} . (\exists \text{topic} . \top)$



Reduction

getting rid of acyclic TBoxes

Expansion of concepts and ABoxes:

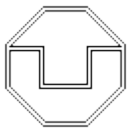
Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, where $\mathcal{T}$ is acyclic, and C a concept description.

The expanded versions $\hat{C}$ and $\hat{\mathcal{A}}$ of C and $\mathcal{A}$ w.r.t. $\mathcal{T}$ are obtained as follows:

- replace all defined concepts occurring in C and $\mathcal{A}$ by their definitions in the expanded version $\hat{\mathcal{T}}$ of $\mathcal{T}$.

Proposition 2.20

1. C is satisfiable w.r.t. $\mathcal{T}$ iff $\hat{C}$ is satisfiable
2. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent iff $(\emptyset, \hat{\mathcal{A}})$ is consistent



Proof: blackboard

Reduction

getting rid of acyclic TBoxes

Expansion of concepts and ABoxes:

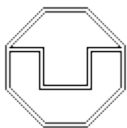
Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, where $\mathcal{T}$ is acyclic, and C a concept description.

The expanded versions $\hat{C}$ and $\hat{\mathcal{A}}$ of C and $\mathcal{A}$ w.r.t. $\mathcal{T}$ are obtained as follows:

- replace all defined concepts occurring in C and $\mathcal{A}$ by their definitions in the expanded version $\hat{\mathcal{T}}$ of $\mathcal{T}$.

Proposition 2.20

1. C is satisfiable w.r.t. $\mathcal{T}$ iff $\hat{C}$ is satisfiable
2. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent iff $(\emptyset, \hat{\mathcal{A}})$ is consistent



Similar reductions exist for the other reasoning problems.

Reduction

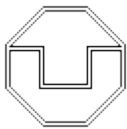
getting rid of the TBox

This reduction is in general **not polynomial**,
since the expanded versions may be **exponential** in the size of $\mathcal{T}$.

$$\begin{aligned} A_0 &\equiv \forall r.A_1 \sqcap \forall s.A_1 \\ A_1 &\equiv \forall r.A_2 \sqcap \forall s.A_2 \\ &\vdots \\ A_{n-1} &\equiv \forall r.A_n \sqcap \forall s.A_n \end{aligned}$$

The size of $\mathcal{T}$ is linear in n ,
but the expansion version $\hat{A}_0$ of A_0 contains A_n 2^n times.

Proof: induction on n



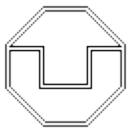
Relationship with First-Order Logic

Reasoning in $\mathcal{ALC}$ can be translated into reasoning in first-order logic:

Lemma 2.21

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, C, D be $\mathcal{ALC}$ -concept descriptions, and a an individual name.

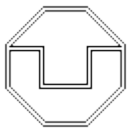
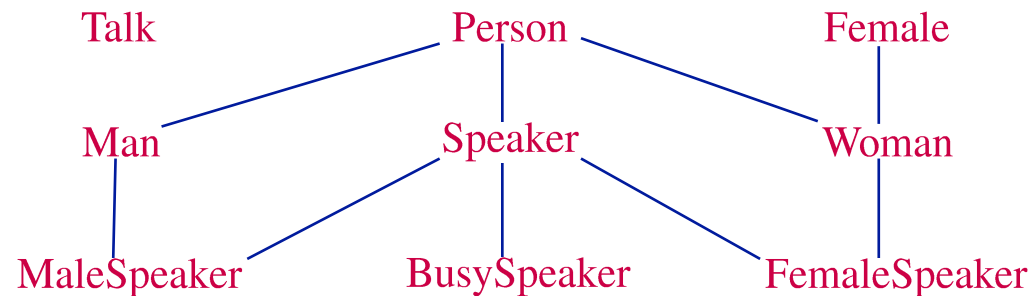
1. $C \sqsubseteq_{\mathcal{T}} D$ iff $\tau(\mathcal{T}) \models \forall x. (\tau_x(C)(x) \rightarrow \tau_x(D)(x))$
2. $\mathcal{K}$ is consistent iff $\tau(\mathcal{K})$ is consistent
3. a is an instance of C w.r.t. $\mathcal{K}$ iff $\tau(\mathcal{K}) \models \tau_x(C)(a)$



Classification

Computing the subsumption hierarchy of all concept names occurring in the TBox.

$\text{Man} \equiv \text{Person} \sqcap \neg \text{Female}$
 $\text{Woman} \equiv \text{Person} \sqcap \text{Female}$
 $\text{MaleSpeaker} \equiv \text{Man} \sqcap \exists \text{gives.Talk}$
 $\text{FemaleSpeaker} \equiv \text{Woman} \sqcap \exists \text{gives.Talk}$
 $\text{Speaker} \equiv \text{FemaleSpeaker} \sqcup \text{MaleSpeaker}$
 $\text{BusySpeaker} \equiv \text{Speaker} \sqcap (\geq 3 \text{ gives.Talks})$



Realization

Computing the most specific concept names in the TBox to which an ABox individual belongs.

$$\begin{aligned}\text{Man} &\equiv \text{Person} \sqcap \neg \text{Female} \\ \text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\ \text{MaleSpeaker} &\equiv \text{Man} \sqcap \exists \text{gives.Talk} \\ \text{FemaleSpeaker} &\equiv \text{Woman} \sqcap \exists \text{gives.Talk} \\ \text{Speaker} &\equiv \text{FemaleSpeaker} \sqcup \text{MaleSpeaker} \\ \text{BusySpeaker} &\equiv \text{Speaker} \sqcap (\geq 3 \text{ gives.Talks})\end{aligned}$$
$$\text{FRANZ} : \text{Man}, \quad (\text{FRANZ}, \text{T1}) : \text{gives}, \\ \text{T1} : \text{Talk}$$

FRANZ is an instance of Man, Speaker, MaleSpeaker.
most specific

